2017

A machine learning approach to the classification of phishing bot accounts within Twitter

Brake, C.

Brake, C. (2017) 'A machine learning approach to the classification of phishing bot accounts within Twitter', The Plymouth Student Scientist, 10(2), p. 208-223.
http://hdl.handle.net/10026.1/14166

The Plymouth Student Scientist
University of Plymouth

*The Plymouth Student Scientist, 2017, **10** (2), 208-223*

A machine learning approach to the classification of phishing bot accounts within Twitter

Christopher James Brake

Project Advisor: Stavros Shiaeles, School of Computing, Electronics and Mathematics, Plymouth University, Drake Circus, Plymouth, PL4 8AA

Abstract

Social network bots are becoming an ever-greater threat to online users. Most studies carried out have looked at bots which generate a lot of tweets known as spam, as these are very common. In recent years research into the area of bots within Twitter has been carried out using machine learning to attempt to find patterns in these accounts to aide with detection. However, limited research has been carried out that focuses on a sub set of Twitter bots which are involved in phishing campaigns which tweet very little to avoid detection. In this project an application was developed that combines a variety of commercial tools with machine learning theory to allow a user to collect and analyse public Twitter data using a neural network. The focus of the project is to try and find patterns in these phishing bots' properties and to use the data collected to train a neural network to recognise these patterns and detect bots. A Twitter crawler was developed that harvests data from the Twitter API and stores it in a graph database. The data is then formatted and normalised by a pre-processor module which is then fed into a neural network. The neural network evaluates the data and creates predictions based on what it has previously learnt, these predictions are then displayed in a graph format within the browser. Experimental results have shown that there is a pattern in the properties of an account, and tests showed a correlation in the friend to follower ratio of bot accounts. With this pattern and other properties of an account, a neural network has been trained to detect bot accounts, with tests showing the neural being able to make predictions for an account with an accuracy of 92%. Whilst these results are still experimental the project has proven that is it possible to detect bots within Twitter using just the properties of an account.

Introduction

Twitter is currently one of the most used social media network platforms, with over 555 million registered accounts and an estimated 200 million active users (Knight-McCord et al., 2016). Twitter's main purpose is that of a microblogging service which allows a user to update their status, known as a tweet (a message or post, limited to 140 characters) to their friends and followers. Twitter also allows for the retweeting of content, this is where a user shares a tweet created by someone else on their own Twitter feed, which can then be viewed by their friends and followers. An important feature of Twitter is its use of hashtags; written as the '#' symbol. Hashtag's are often single words or phrases which are used to create an index of keywords or topics that users can follow. A user is then able to search Twitter for specific hashtags and see what Tweets have used that hashtag. Twitter creates an online social structure consisting of clusters of interconnecting people, with celebrities having huge numbers of followers.

Although Twitter has been designed as a place for people to share their views, thoughts, opinions and much more, some have taken this functionality and abused it for their own personal gain or malicious intent. The majority of such use is carried out by using robotic accounts, which are commonly known as bots. A bot account is an account controlled by a computer program that has been coded to interact with Twitter's API to perform automated tasks. A bot can be used for a variety of relatively harmless, sometimes useful tasks such as targeted marketing or spreading of news articles. Some accounts are designed for spamming and link redirection, where the creator is paid for each person they redirect to a specific site (Subrahmanian et al., 2016). There is then the more malicious pursuits such as phishing campaigns and links to malware (Chu et al., 2011). A study carried out by Zhang and Paxson in 2011 estimated that 16% of all Twitter accounts showed high signs of being automated (Zhang and Paxson, 2011).

Bot accounts in Twitter are most commonly used for the creation of spam tweets and the spreading of articles. This has led to a lot of research and development into spam filter detection and utilising classification algorithms and machine learning to determine if an account is a bot account or not (Wang, 2010). Twitter has been trying to tackle the growing number of automated accounts amongst its millions of legitimate users. This has involved using sophisticated spam filters which analyse the contents of tweets, evaluating if a posted link is malicious or not. Twitter has also implemented a 'Report as spam' feature which allows Twitter users to report accounts or tweets they believe to be spam. With Twitter being such a huge public platform with accounts across the globe, the rise of malicious bot accounts possesses the potential to become a huge cyber security problem.

The main purpose of this project is to try and build an application that makes use of machine learning and artificial intelligence (AI) theories and to combine them with commercial technologies to be able to find and display bot accounts within Twitter. The field of AI has progressed much in recent years, leading to it becoming the next era of IT, offering new possibilities for a variety of areas. With cyber security becoming prevalent in daily life it is time that cyber security professionals start to utilise AI to provide new ways of protecting people and infrastructure online. If developed properly an AI could be created that would be able to independently monitor Twitter and seek out illegitimate accounts creating a safe environment for users. This would be a more efficient alternative to the spam filters that are currently being used as the AI would in theory not need human intervention, it would continue to learn and adapt. However,

this brings up several issues, one around privacy if the AI were to store any user data that it needed to analyse. There are then ethical and professional issues surrounding the misclassification of a Twitter account what knock on effects this would have.

An aim of this project is to be able to build a neural network that can detect bot accounts. In particular, the bot accounts which are part of phishing campaigns that often deploy what are known as 'click bait' bots. These bots aim to attract users into clicking malicious links by displaying provocative images, these links will often contain malware or take a user to a site where they are deceived into giving person details which can later be used for fraud. The approach that will be taken will involve looking past the content of a tweet and an accounts behaviour but will instead involve looking at properties of an account and trying to find a pattern in the data. If a pattern does exist and these kinds of bots consist of similar properties then it will be tested whether an AI can use this knowledge to learn to classify bots.

This project plans to employ machine learning techniques to identify the varying different types of accounts that are found within Twitter, with the aim of looking past spam bots and finding bot accounts that have been designed to behave maliciously and avoid spam filters. This report will first look at the current research in the topic area, this will then be followed by an overview of all the different sections of the application. The report will then detail the results that were found which leads into a discussion about what the project has found. Finally, the report will describe potential future research and development and then draws conclusions from the work that has been carried out.

Related work

The detection of bot accounts within Twitter has been an area of research that has gained more interest over the past few years. A lot of the attempts to detect bot accounts within Twitter involve trying to detect spam and looking at the content within tweets using sophisticated spam filters. A method proposed by Wang (2010) outlined a machine learning approach to detecting spambots. His approach involved using a collection of classification methods including a neural network, to generate a spam probability for an account. The approach taken by Wang focuses on detecting spambots which are constantly tweeting and following random accounts. Whilst the methods used were successful at detecting spambots he fails to consider other bot accounts within Twitter who tweet very little to avoid spam filters. These kinds of accounts post a few initial tweets that contain malicious links, after that these types of accounts are then "liking" or retweeting content to appear active. They also follow lots of users with the aim of getting their malicious links clicked. Due to the way he has trained his neural network, these kinds of accounts would not be detected. However, they still pose a serious threat to users.

Chu et al. (2011) built a Twitter bot classification system that is based on machine learning techniques. The approach taken is more comprehensives than other methods looking at just spam as they also look at other features of a Twitter account, these included aspects such as URL ratio, entropy rate of tweeting and device tweeted from. Whilst this approach is very effective at detecting bot accounts it also requires a lot of input data to try and determine if an account is legitimate or not.

This project aims to use machine learning to try and discover patterns in diverse types of bot accounts based on a set of properties of an account. This would allow for the detection of illegitimate accounts that do not only just generate spam but also have

other malicious intents. It would also mean that the application could make these classifications based on a much smaller subset of evaluation data, which would make the process faster.

Proposed method

The application that was developed during the project consists of five independent modules. This section of the report will describe how each module was developed and the technologies each one utilises to function.

Data collection

To collect data that would be used for training and testing of the neural network, an application needed to be built to communicate with Twitters REST API. PyTrawler is a Python based Twitter crawler that is designed to collect data from public accounts, it consists of two Python libraries; Tweepy for communicating with the Twitter API's and Py2neo for communicating with Neo4j databases. The issue with using the REST API is that an application can only make so many requests to the API in a certain time window, as stated in the Twitter Developer policy (Twitter, 2016). This means that speed of harvesting data is limited. Once the API request threshold is reached a 15-minute waiting period must then take place. PyTrawler has been built to handle this and will wait once it reaches the API limit. The application has also been designed to handle the loss of internet connectivity. This was implemented to allow the application to continue crawling for long periods without having to worry about resetting it.

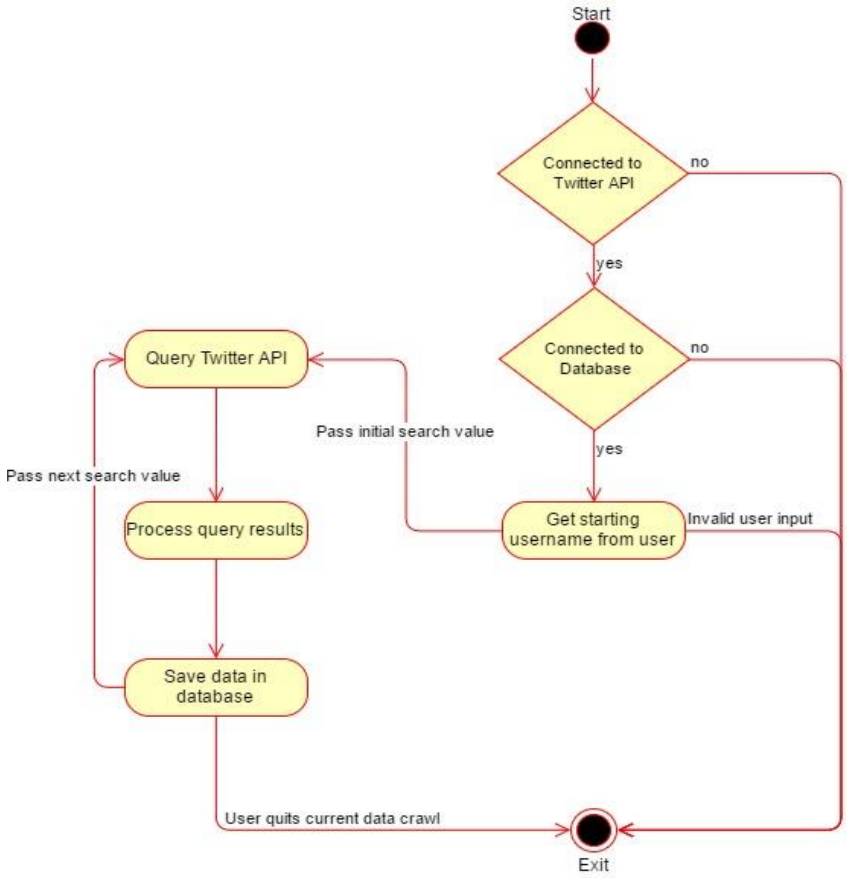

Figure 1 - PyTrawler process flow diagram

Data storage

A graph database format is used to store the data collected by PyTrawler, (Neo4j, 2017) is the technology used within the project. A graph database was implemented as it provides a data structure that fits social networks and allows for quick querying and traversal through the data. Neo4j was selected because it provides a simple to setup and install graph database that offers a browser front end which visualises the data as seen in Figure 2. Once a Neo4j instance is setup, PyTrawler can be configured to connect with the database and will then be able to store the data it harvests.

Figure 2 - PyTrawler data in Neo4j

Data processing

With the raw data being collected by PyTrawler and stored in a Neo4j instance, the data had to be processed to create data sets that could be used by the neural network. The two issues that had to be tackled were formatting and normalising the data, the solution involved creating a pre-processor module as seen in Figure 3. The pre-processor consists of two Java classes, one for connecting to the Neo4j instance and collecting the data based on a cipher query. The second class carries out all the formatting and normalising, achieving this by utilising the technology of Apache Spark (Apache, 2017).

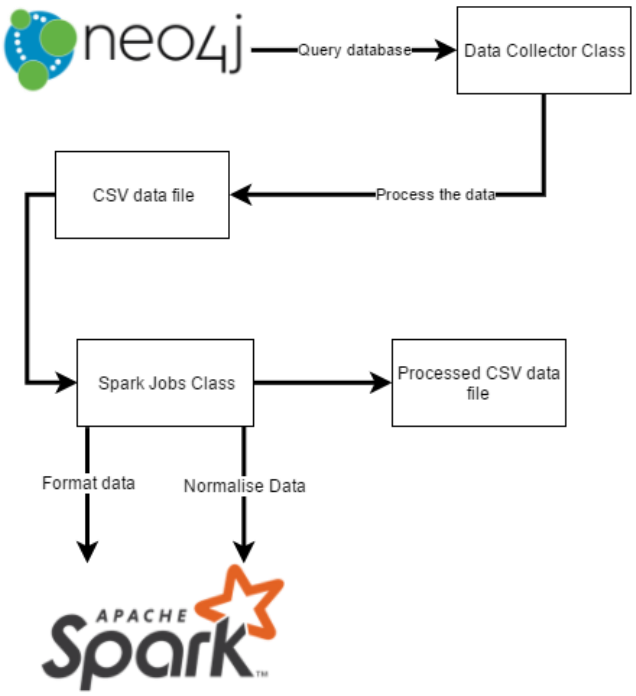

Figure 3 - Pre-processor flow diagram

Machine learning

The machine learning aspect of the project involved the creation of an application named BotSpot which is built upon a machine learning library written in Java called DeepLearing4j (DL4J) (Gibson and Nicholson, 2017). A simple classification network was developed using the machine learning theory of Multilayer Perceptron's (MLP). DL4J allows for the configuration and building of an MLP, within the configuration, aspects such as the learning rate, activation function and number of hidden layers are defined.

From the pre-processor module, a data set was created, from this data set a training and test set were created. Creating the data sets involved a manual process of evaluating the data and checking the Twitter account to determine whether it was a bot account or not. The data within the training or test set was then given a label of either 'HUMAN' or 'BOT'. The neural network model is then given the training data which it analyses and learns from based on the labels. It then uses the test data set to evaluate itself, giving results as seen in Figure 4. The initial training of the neural network used a training set consisting of 760 rows of data 40 of which were bot accounts. The initial neural network was not very accurate and failed to classify any bot accounts.

```
Examples labeled as 0 classified by model as 0: 150 times
Examples labeled as 1 classified by model as 0: 12 times

Warning: class 1 was never predicted by the model. This class was excluded from the average precision

========================Scores========================================
 Accuracy:        0.9259
 Precision:       0.9259
 Recall:          0.5
 F1 Score:        0.6494
========================================================================
```

Figure 4 - Neural network initial training results

Configuring the neural network

After the results of the initial training, it was clear that the neural network was not configured correctly as it was completely failing to identify bot accounts within the test data. To improve the accuracy of the neural network some of the variables were changed. After each change the newly configured neural network was then tested and the results compared; the results of which can be seen in Figure 5. The three variables that were changed in the configuration were the learning rate, the number of passes through the data and the number of hidden nodes. From the results of the tests it was found that a small increase in the number of hidden nodes had no effect on the accuracy of neural network, further tests would need to be carried out to ensure that this was conclusive. The learning rate was the variable that had the most significant effect on the accuracy of the neural network. When the learning rate was increased to 0.5 the accuracy of the neural network decreased to just below 18%, with the network predicting that most of the test data were bots. The learning rate was then lowered to 0.05 where the accuracy increased to around 50%.

From the tests, it was seen that the number of passes through the data also effected the accuracy of the neural network. When the number of passes was increased from 30 to 50 the accuracy was raised from 45% to 56%. The highest number of passes used was 200, more would have been tested but due to the limitation of the computing power the testing was restricted. From the various tests that were carried out it was found that the optimum configuration was a learning rate of 0.02 and 200 passes. Although there were not enough tests carried out to conclusively prove that a continued increase in passes would proceed to enhance the accuracy, or whether the number of passes would hit a point where it no longer had an effect.

To try and further improve the accuracy of the neural network a new data set was created consisting of data which hadn't been classified. A portion of the predictions made by the neural network on the data were then analysed manually and added to a new training data set of 100 rows which were then correctly labelled. The neural network was then retrained using the new training set resulting in a more accurate neural network.

Evaluating and labelling the data

Within BotSpot there is an evaluation class, its purpose is to allow for the analysis and evaluation of unclassified data using a neural network. The class reads in data that has been generated by the pre-processor and splits off the ID of an account and stores it to use later. The reason the ID has to be split from the data is because the neural network can't process the it. Once this is done the evaluator class reconstructs a previously trained neural network, it then uses this neural network to evaluate the data. The neural network then creates two predictions for each row of data; a human percentage and a bot percentage. After this the class then appends both the predictions and the user ID back to the data and outputs it all as a CSV file. The evaluated data is then passed to a labelling class. The purpose of this class is to update the data within the Neo4j database with a 'Bot' or 'Human' label based on the predictions made by the neural network. The class works by first extracting the user ID and the two percentages given to the data by the neural network. The first percentage is how sure the neural network is that the account is human and the second for how sure it is that

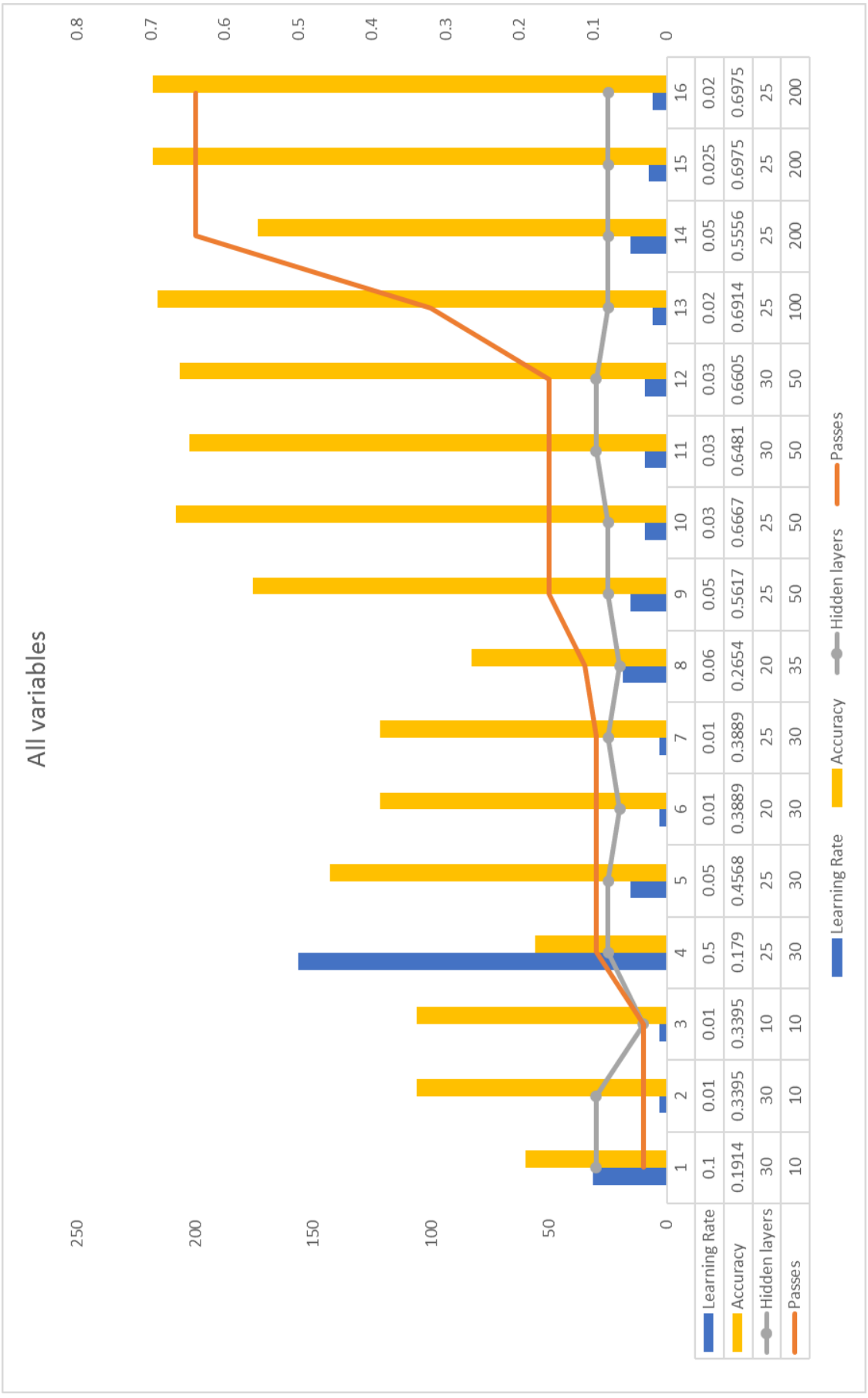

Figure 5 - Results from configuring the neural network

the account is a bot. The two percentages are then evaluated, depending on which is higher determines which query will then be executed on the Neo4j database to update the node.

Data visualisation

A web front end was developed to display the Twitter data that has been evaluated and labelled by the machine learning module of the project. The main aim of the front end is to provide an easy to understand visualisation of the data which clearly highlights which accounts are potential bots.

To achieve this a web server had to be developed which needed to be able to display the web page and carry out the querying and processing of data from Neo4j. Bottle was selected as the framework to be used. It is a micro web framework written in Python that is lightweight and simple to use (Bottle, 2017). Bottle works by allowing the developer to define functions that are called when URL's are requested by the browser front end. The Bottle web server then automatically handles the requests and executes any function within the Python code that has been defined with that URL.

The webpage itself uses Alchemy (2017) which is a JavaScript library that is built on top of the powerful D3 Javascript library that is widely used for displaying graph data (Bostock, 2017). Alchemy is designed to do all the heavily lifting required to interact with the D3 library. Alchemy.js requires that the data supplied to it to be in a specific GraphJSON format. A Neo4j cipher query was developed that collected the desired data and stored the results in the correct format. The Bottle server then executed this cipher query and saved the results into a JSON which is then read in by the Alchemy.js configuration.

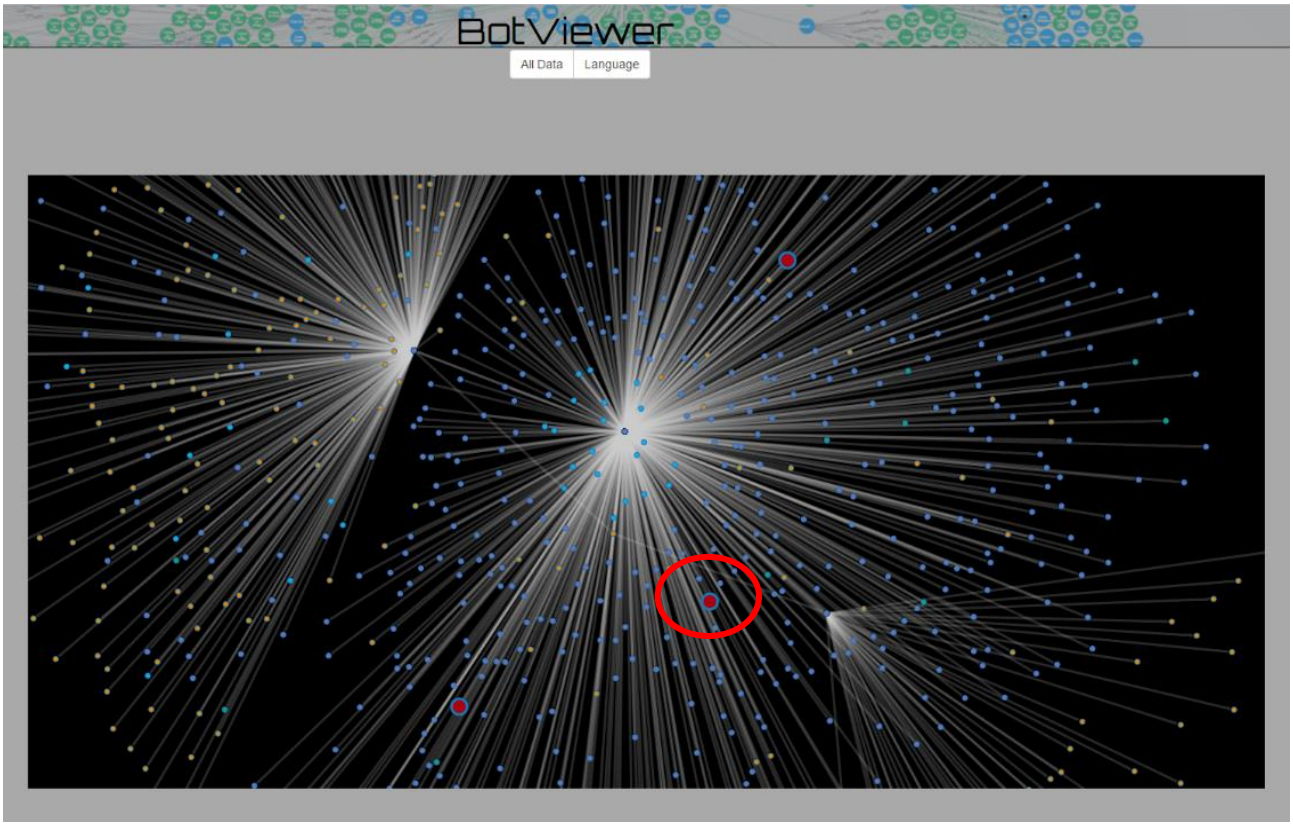

Figure 6 - BotViewer graph representation of the neural network predictions

Figure 6 shows a graphical representation of the data that has been harvested by PyTrawler, each node is an object such as a user or a tweet. The colours indicate what each node represents, for example orange nodes are friends of the user they are connected to. The links that connect each node describe the relationship that exists between them, for example a user node will be connected to a tweet with the link 'tweeted'. BotViewer also displays the predictions made by the neural network, usually a user is purple, if a node has been evaluated by the neural network it will be given a new label; green if it is deemed human and red if it is a bot. A bot can be seen in the red circle in Figure 6, it has been made larger than other nodes to ensure they stand out within the graph.

End project architecture

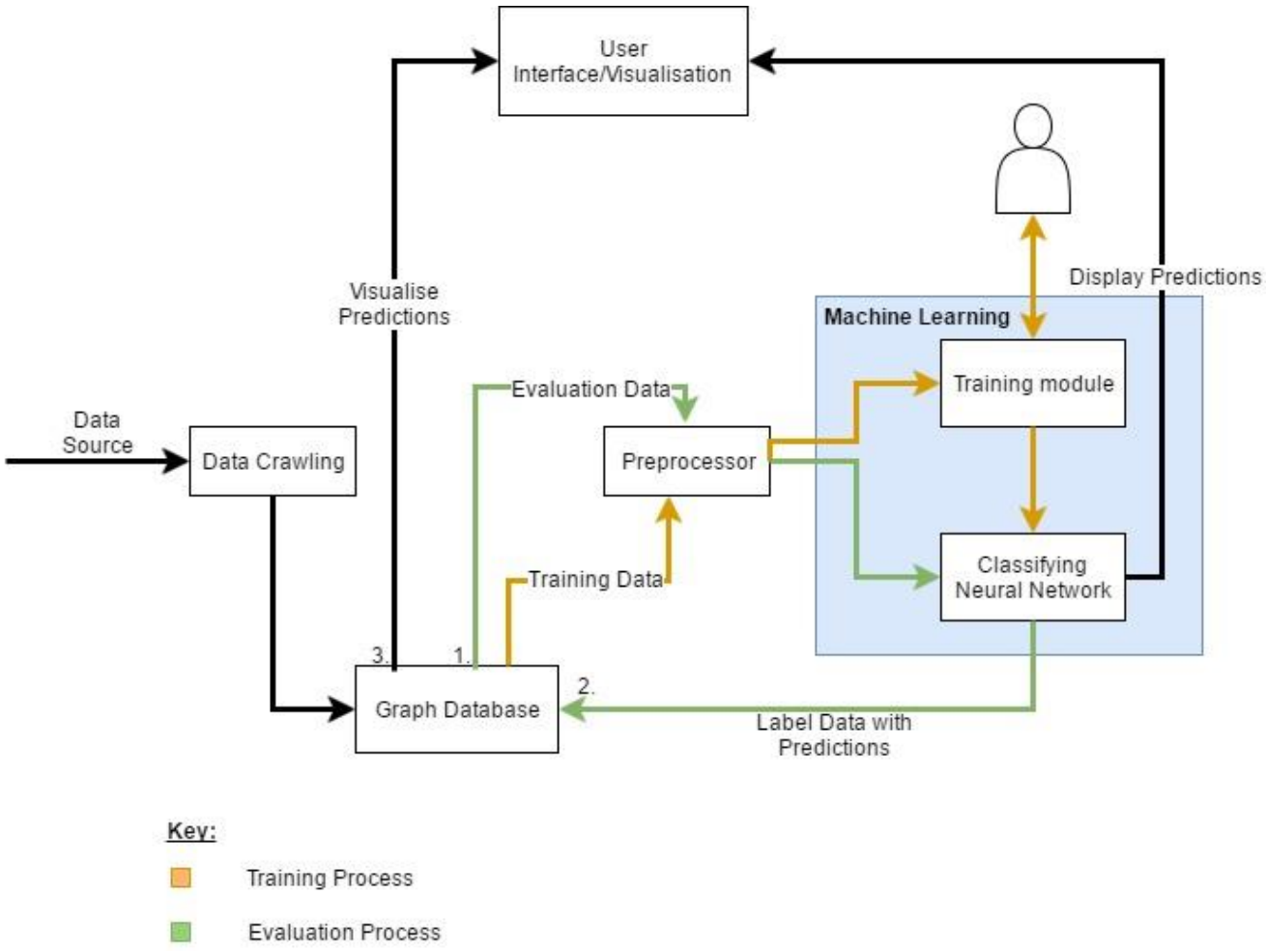

Figure 7 - Final project architecture

Results

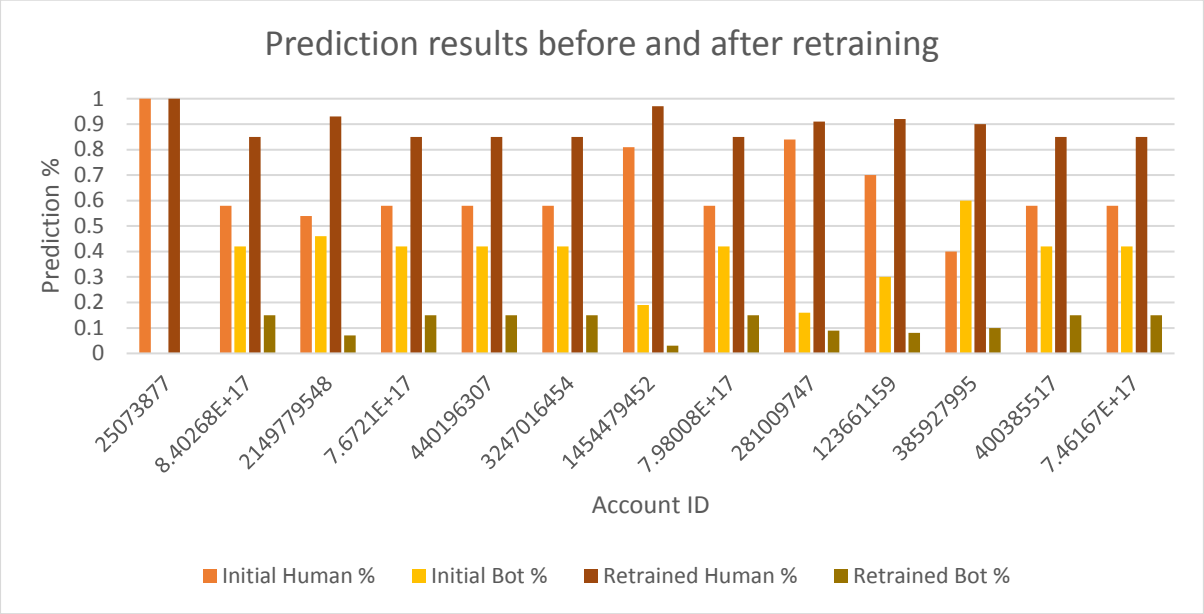

Figure 8 - Results of the predictions made by the neural network

Figure 8 shows a sample of the predictions made by the neural network on the same data set before and after training. The results show how the neural networks confidence in predicting an account as being a human increased. An interesting outcome from the results is how the neural network learnt to always label a Twitter account that is marked as 'verified' as human, this can be seen by the results of account ID 25073877 where the human prediction was 100% both before and after retraining.

In this sample, the retrained neural network doesn't have any bot account predictions that are over 0.2 whereas the initial predictions for bot accounts were as high as 0.6. This sharp change in the neural networks confidence that these accounts were bots is caused by initial misclassification. The dataset that was used for retraining the neural network was generated from a sub set of the data the initial predictions were made on. Accounts which were initially classified incorrectly by the neural network were then labelled correctly in the new training data. When the neural network was retrained and fed the same data set again its predictions for the misclassified accounts changed. This demonstrates how the neural network can learn quickly from small amounts of data. Retraining the network using only 100 rows of data meant that the networks accuracy went from 69% to 92%.

During the creation of the training data set it was discovered that the bots that had been found manually by sifting through the data collected by PyTrawler all had a similar normalised friend to follower ratio (see Figure 9).

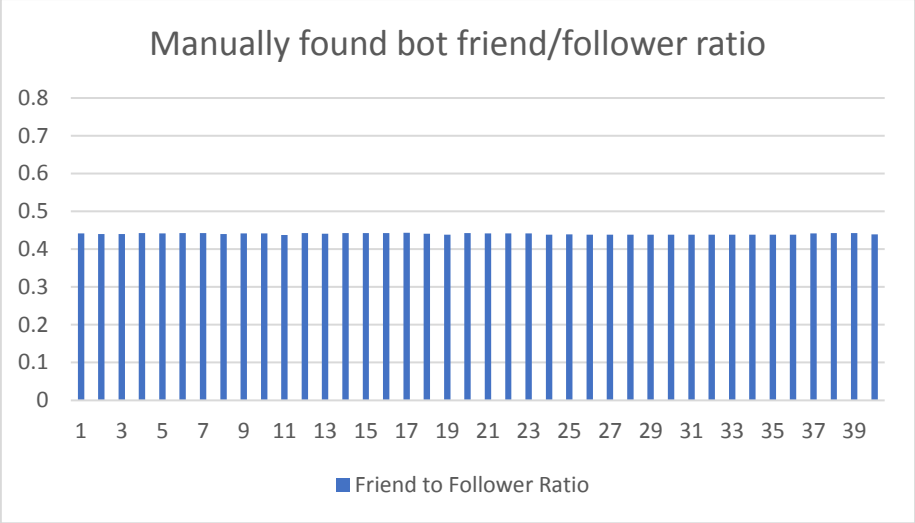

Figure 9 - Friend to follower ratio of manually found bots

The friend to follower ratio of all the bots used within the training data were just below a value of 0.45. This means that the bot accounts have a higher number of people they follow than the number of people following them back an example account can be seen in Figure 10.

Figure 10 - Example bot account within Twitter

A similar trend can be seen in the predictions made by the neural network after it was retrained. Figure 11 shows a sample of the data that the neural network predicted were bot accounts and their corresponding friend to follower ratio. These potential bots have not been verified but the results show that the majority of the potential bots all have a similar friend to follower ratio as the bots from the training data. The average friend to follower ratio of the bots predicted by the neural network is 0.51. A contributing factor for the increase in the average friend to follower ratio is the difference in the data sizes. The training set consisted of 760 rows, the uncategorised data predictions were made on a data set that consisted of 100,000 rows.

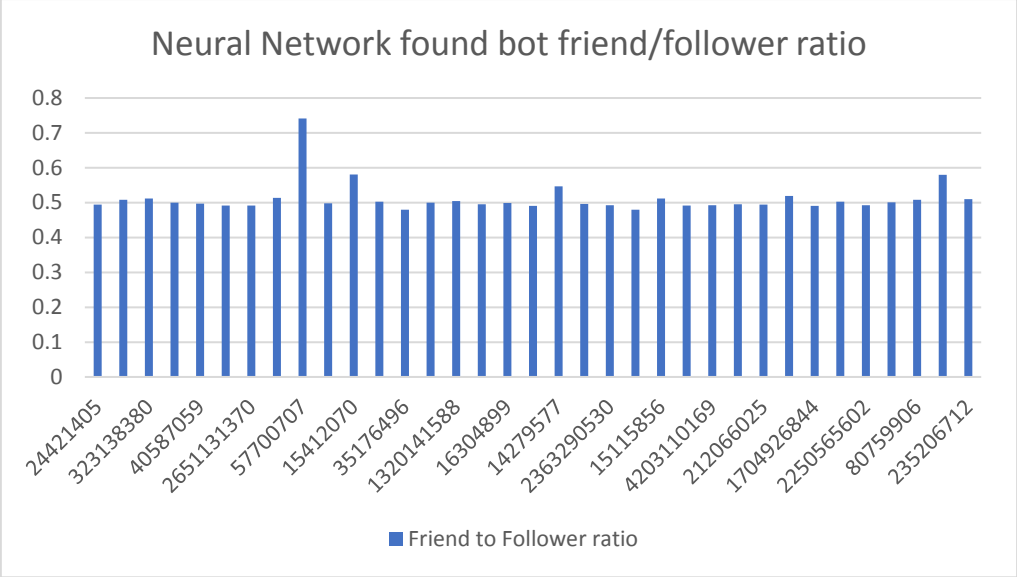

Figure 11 - Friend to follower ratio of bots predicted by the neural network

Discussion

Overall the project went well. It is a collection of different modules, which when combined, allow a user to analyse Twitter accounts and, through machine learning, predict how likely an account is to be a bot. This data can then be displayed in the browser as a graph database that provides an easy to understand interface for viewing the predictions made by the neural network. The results from the training of the neural network show that it can be trained to detect bots using only the properties of an account, obtaining an accuracy score of 92% after one round of retraining. The results suggest that there is a common pattern in click bait bots used in phishing campaigns, notably that these kinds of accounts have similar friend to follower ratios.

Despite the neural network evaluating itself as having an accuracy of 92% it is not as accurate and precise as it would need to be if it were to ever end up in a commercial environment. There are several reasons why the neural network is still inaccurate, the main reason being that there wasn't enough data used to train it. The initial training set consisted of 760 rows of data, 40 of which were known bot accounts. There was also a small test dataset which had 150 rows in it, 12 of which were known bot accounts. Having such small datasets means that the neural network is only getting to learn about a tiny fraction of the accounts that make up Twitter. The reason the datasets were so small is due to how time consuming it is to make them, each dataset had to be made manually by checking each account within Twitter and clarifying whether it was legitimate or not. This is one of the downsides to the supervised learning approach taken within the project but it was necessary to ensure the neural network was trained to find the correct bot accounts.

The statistical results involving the friend to follower ratio of a bot account indicate that there is a pattern between these kinds of bots. The data suggests that a phishing bot account will only follow a certain number of accounts at any one time which aligns with Chu et al. (2011) who also found that bots follow many users but have far less users follow them back. They also state how bots will often unfollow an account which hasn't followed them back; this is done to circumvent Twitter's limit on the ratio of friends to

followers imposed to try and tackle bots. Carrying out the practice of unfollowing an account allows the bot to go on to follow another account, of which many follow them back, allowing that user to view the bots' malicious tweets.

One of the aims of this project was to see if it was possible to detect bot accounts based purely on the properties of an account. The data collected suggest that there is a pattern in the bot accounts that are used within phishing campaigns, most noticeably around the friend to follower ratio. However, there is a small percentage of legitimate accounts having similar properties to that of a bot account, this was mainly seen in accounts where people have followed a lot of people but don't have many followers themselves. With these kinds of legitimate accounts causing false positives within the data other fields would need to be used that could allow the neural network to discern if it is a bot or human account. Whilst this easy enough to accomplish, the more fields that are required for evaluation the more data that needs to be collected and the more data there is to format and then analyse, inevitably this will slow the prediction process down.

Future work

With the project consisting of a collection of different modules such that each can be classed as their own independent applications, there is lots of room for further expansion and research. PyTrawler is currently only a command line tool. To make it a more user-friendly application a graphical user interface (GUI) could be developed. The GUI would make using PyTrawler easier as it could provide simple functionality for customising which Twitter accounts to crawl and therefore tailor what data is collected. Another area of expansion for PyTrawler would be to integrate connectivity to the Twitter streaming API, providing the user with the ability to collect live Tweet data. PyTrawler could be built to query both API's and collect enriched live data for the machine learning module to ingest. The data collection aspect of the pre-processor is currently just a basic Java class that carries out predefined cipher queries and collects the results from a Neo4j instance. To make it a more usable application a GUI could be developed that allows a user to define custom queries with the help of a query builder. The query builder could provide an auto-complete feature to make generating cipher queries easier for those new to the syntax. In the future, the pre-processor could be developed to handle the processing of data from different sources other than Twitter, this could include sites such as Facebook and Pinterest. The GUI could offer the ability to point at different databases which would allow for the processing of different data sources. BotSpot has the potential to be expanded greatly. It currently only consists of one simple classification MLP, if developed further it could be built to have multiple, more complex neural networks that would provide more accurate and precise results. It could potentially be built to use other applications such as Google's image search API to try and find bot accounts that use the same generic profile images and try to determine links between these accounts. BotViewer provides very limited visualisations and functionality in its current state, with further development it could be built to display graphs based on custom queries as well as offer filtering and searching of the data. If it were to be developed into a live monitoring tool it could provide updates of potential bots that have been labelled by the machine learning module.

Conclusions

This project set out to try and build an application that utilised machine learning to try and detect bot accounts within Twitter. On a basic level, this has been achieved, the

application can harvest, process then visualise Twitter data and highlight what it believes to be bot accounts. The project has met all the objectives that were originally set out during its design, not necessarily in the way that was planned. The final product is a collection of different modules that provide a user with the functionality to analyse Twitter data, although each module is far from complete. When considered, the time frame of the project imposed constraints on the development. Ultimately leading to only core functionality being achieved, more time would be needed to build an application that more accurately and precisely detect bots within Twitter. However, the project has laid the groundwork for such an idea, with bots becoming more of a threat every day, applications such as this will be needed. Machine learning and AI will play a key role in the future of fighting cyber criminals and protecting users on the internet. The machine learning module has proved that it is possible, through the analysis of account data, to detect bot accounts within Twitter by only looking at a small number of account properties. Despite the neural network not being 100% accurate it was able to learn and improve itself with minor supervision. The data used within this project is but a fraction of the amount needed to properly train the neural network.

Acknowledgements

I would like to extend my sincere gratitude to my project supervisor, Stavros Shiaeles, for his overwhelming support, guidance, and enthusiasm throughout this project. He has pushed me to achieve my best. I would also like to thank the Deeplearning4j Development team, without whose library and documentation, would mean that this project would lack serious functionality. Finally, I would like to thank my friends and family for they support and encouragement through the sometimes-difficult challenges and circumstances that were faced throughout this project.

Reference list

Adeli, H. and Hung, S. (1995). *Machine learning*. New York: John Wiley & Sons.

Alchemy (2017). Alchemy.js. [online] Graphalchemist.github.io. Available at: http://graphalchemist.github.io/Alchemy/#/ [Accessed 8 May 2017].

Apache Spark (2017). Apache Spark™ - Lightning-Fast Cluster Computing. [online] Spark.apache.org. Available at: https://spark.apache.org/ [Accessed 8 May 2017].

Bostock, M. (2017). D3.js - Data-Driven Documents. [online] D3js.org. Available at: https://d3js.org/ [Accessed 8 May 2017].

Bottle (2017). Bottle: Python Web Framework — Bottle 0.13-dev documentation. [online] Bottlepy.org. Available at: http://bottlepy.org/docs/dev/ [Accessed 8 May 2017].

Chu, Z., Gianvecchio, S., Wang, H., & Jajodia, S. (2012). Detecting automation of twitter accounts: Are you a human, bot, or cyborg? IEEE Transactions on Dependable and Secure Computing, 9(6), 811-824.Gibson, A. and Nicholson, C. (2017). Deeplearning4j: Open-source, Distributed Deep Learning for the JVM. [online] Deeplearning4j.org. Available at: https://deeplearning4j.org/ [Accessed 8 May 2017].

Gibson, A. and Nicholson, C. (2017). ND4J: N-Dimensional Arrays for Java - N-Dimensional Scientific Computing for Java. [online] Nd4j.org. Available at: http://nd4j.org/index.html [Accessed 8 May 2017].

Grier, C., Thomas, K., Paxson, V., & Zhang, M. (2010, October). @ spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security* (pp. 27-37). ACM.

Jones, M. (2005). AI application programming. 1st ed. Hingham (Mass.): Charles River Media.

Knight-McCord, J., Cleary, D., Grant, N., Herron, A., Lacey, T., Livingston, T., & Emanuel, R. (2016). What social media sites do college students use most? Journal of Undergraduate Ethnic Minority Psychology, 2, 21.

Neo4j (2017). Neo4j, the world's leading graph database - Neo4j Graph Database. [online] Neo4j Graph Database. Available at: https://neo4j.com/ [Accessed 17 May 2017].

OpenBLAS (2017). OpenBLAS: An optimized BLAS library. [online] Openblas.net. Available at: http://www.openblas.net/ [Accessed 8 May 2017].

Porter, B., Zyl, J. and Lamy, O. (2017). Maven – Welcome to Apache Maven. [online] Maven.apache.org. Available at: https://maven.apache.org/ [Accessed 27 Apr. 2017].

Subrahmanian, V. S., Azaria, A., Durst, S., Kagan, V., Galstyan, A., Lerman, K. & Menczer, F. (2016). The DARPA Twitter bot challenge. Computer, 49(6), 38-46.

Thomas, K., Grier, C., Song, D., & Paxson, V. (2011, November). Suspended accounts in retrospect: an analysis of twitter spam. In Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference (pp. 243-258). ACM.

Twitter (2016). Developer Agreement & Policy. [online] Available at: https://dev.twitter.com/overview/terms/agreement-and-policy [Accessed 8 May 2017].

Wang, A. H. (2010). Detecting Spam Bots in Online Social Networking Sites: A Machine Learning Approach. DBSec, 10, 335-342.

Zhang, C., & Paxson, V. (2011). Detecting and analyzing automated activity on twitter. In Passive and Active Measurement (pp. 102-111). Springer Berlin/Heidelberg.

Zhang, Q. J., Gupta, K. C., & Devabhaktuni, V. K. (2003). Artificial neural networks for RF and micro-wave design-from theory to practice. IEEE transactions on microwave theory and techniques, 51(4), 1339-1350.